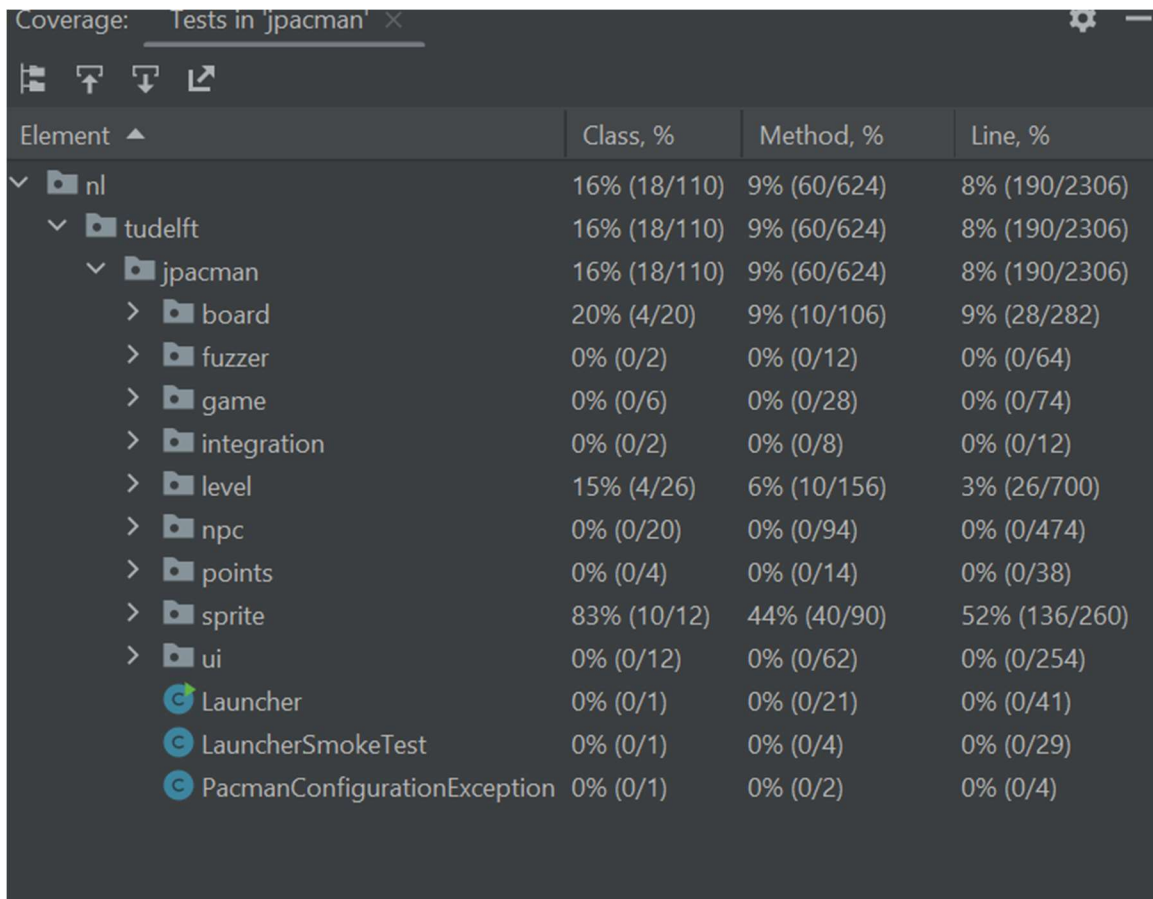


JaCoCo Report on Jpacman

For the Jpacman repository I created three unit tests. The classes named are:


src/main/java/nl/tudelft/jpacman/board/Board.java/squareAt	Dylan Cruz
src/main/java/nl/tudelft/jpacman/board/Board.java/getHeight	Dylan Cruz
src/main/java/nl/tudelft/jpacman/board/Square.java/put	Dylan Cruz

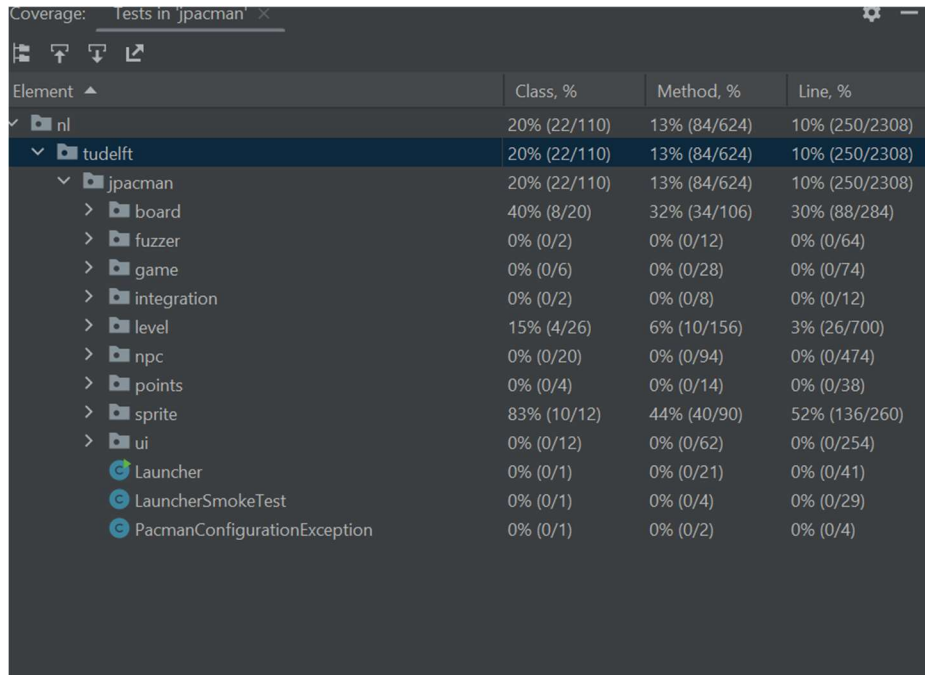
Before I implemented my unit tests the coverage on the IntelliJ window showed 16% class coverage, 9% method coverage, and 8% line coverage.



Element ▲	Class, %	Method, %	Line, %
▼ nl	16% (18/110)	9% (60/624)	8% (190/2306)
▼ tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
▼ jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
> board	20% (4/20)	9% (10/106)	9% (28/282)
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/6)	0% (0/28)	0% (0/74)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	15% (4/26)	6% (10/156)	3% (26/700)
> npc	0% (0/20)	0% (0/94)	0% (0/474)
> points	0% (0/4)	0% (0/14)	0% (0/38)
> sprite	83% (10/12)	44% (40/90)	52% (136/260)
> ui	0% (0/12)	0% (0/62)	0% (0/254)
🔄 Launcher	0% (0/1)	0% (0/21)	0% (0/41)
🔄 LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
🔄 PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

After I implemented my unit tests there was 20% class coverage, 13% method coverage, and 10% line coverage.

Coverage: Tests in 'jpacman' 



The screenshot shows the IntelliJ Coverage tool interface. The 'Element' tree on the left is expanded to show the 'jpacman' package. The table on the right displays coverage percentages for classes, methods, and lines. The 'jpacman' package itself shows 20% class coverage, 13% method coverage, and 10% line coverage. Sub-packages like 'board', 'fuzzer', 'game', 'integration', 'level', 'npc', 'points', 'sprite', and 'ui' show varying levels of coverage, with 'sprite' having the highest coverage at 83% for classes, 44% for methods, and 52% for lines.

Element	Class, %	Method, %	Line, %
nl	20% (22/110)	13% (84/624)	10% (250/2308)
tudelft	20% (22/110)	13% (84/624)	10% (250/2308)
jpacman	20% (22/110)	13% (84/624)	10% (250/2308)
board	40% (8/20)	32% (34/106)	30% (88/284)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4/26)	6% (10/156)	3% (26/700)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

However in the JaCoCo coverage report it shows there was more coverage that IntelliJ didn't.

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
nl.tudelft.jpacman.level	<div><div></div></div>	67%	<div><div></div></div>	57%	74 155	104 344	21 69	4 12
nl.tudelft.jpacman.npc.ghost	<div><div></div></div>	71%	<div><div></div></div>	55%	56 105	43 181	5 34	0 8
nl.tudelft.jpacman.ui	<div><div></div></div>	77%	<div><div></div></div>	47%	54 86	21 144	7 31	0 6
default	<div><div></div></div>	0%	<div><div></div></div>	0%	12 12	21 21	5 5	1 1
nl.tudelft.jpacman.board	<div><div></div></div>	86%	<div><div></div></div>	58%	44 93	2 110	0 40	0 7
nl.tudelft.jpacman.sprite	<div><div></div></div>	86%	<div><div></div></div>	59%	30 70	11 113	5 38	0 5
nl.tudelft.jpacman	<div><div></div></div>	69%	<div><div></div></div>	25%	12 30	18 52	6 24	1 2
nl.tudelft.jpacman.points	<div><div></div></div>	60%	<div><div></div></div>	75%	1 11	5 21	0 9	0 2
nl.tudelft.jpacman.game	<div><div></div></div>	87%	<div><div></div></div>	60%	10 24	4 45	2 14	0 3
nl.tudelft.jpacman.npc	<div><div></div></div>	100%	<div><div></div></div>	n/a	0 4	0 8	0 4	0 1
Total	1,213 of 4,694	74%	293 of 637	54%	293 590	229 1,039	51 268	6 47

Created with JaCoCo 0.8.3.201901230119

JaCoCo evaluates coverage differently from IntelliJ. JaCoCo shows a more detailed example of what branches were covered and what branches were missed. IntelliJ shows either good coverage or bad coverage along with how many times the code was run.

Though the IntelliJ coverage report is sufficient enough to evaluate the tests of the project, the JaCoCo visuals makes the report easier to understand for people that aren't familiar with technical reports.

Code Snippets:

SquaresAt():

```
class atSquareTest {  
  
    no usages  
    @org.junit.jupiter.api.Test  
    void squareAt() {  
        Square[][] temp = {  
            {new BasicSquare(), new BasicSquare()},  
            {new BasicSquare(), new BasicSquare()}  
        };  
        Board tempBoard = new Board(temp);  
        assertThat(tempBoard.squareAt(x: 1, y: 1)).isEqualTo(temp[1][1]);  
    }  
}
```

getHeight():

```
class getHeightTest {  
  
    no usages  
    @org.junit.jupiter.api.Test  
    void getHeight() {  
        Square[][] temp = {  
            {new BasicSquare(), new BasicSquare()},  
            {new BasicSquare(), new BasicSquare()}  
        };  
        Board tempBoard = new Board(temp);  
        assertThat(tempBoard.getHeight()).isEqualTo(2);  
    }  
}
```

Put():

```
class putTest {  
  
    no usages  
    @org.junit.jupiter.api.Test  
    void put() {  
        Unit temp = new Unit() {  
            @Override  
            public Sprite getSprite() {  
                return null;  
            }  
        };  
        Square square = new BasicSquare();  
        square.put(temp);  
        assertThat(square.getOccupants()).contains(temp);  
    }  
}
```

Git Repository: <https://github.com/CruzG-Dylan/cs472-team6>